> **Note**
>
> Starting with version 2.4.8, the `SSLCertificateChainFile` directive is deprecated. Instead, you are requested to provide all certificates in the file pointed to by the `SSLCertificateFile` directive. This change was probably driven by the fact that more sites want to use multikey deployments (e.g., RSA and ECDSA at the same time) and that each key might require a different certificate chain.

Not configuring the entire certificate chain correctly is a frequent mistake that causes certificate warnings for connecting clients. To avoid this problem, always follow the instructions provided by your CA. When renewing a certificate, make sure you use the new intermediate certificates provided; the old ones might no longer be appropriate.

> **Note**
>
> The example in this section assumes that your private key is not protected with a passphrase. I recommend that keys are created and backed up with a passphrase but deployed without a passphrase on the server. If you want to use protected keys, you will have to use the `SSLPassPhaseDialog` directive to interface Apache with an external program that will provide the passphrase every time it is needed.

Configuring Multiple Keys

It's not widely known that Apache allows secure sites to use more than one type of TLS key. This facility, which had originally been designed to allow sites to deploy RSA and DSA keys in parallel, is virtually unused because DSA faded into obscurity for web server keys.

These days, there is a lot of discussion about deploying ECDSA keys in order to improve handshake performance. In parallel, there is a desire to migrate certificate signatures to SHA2, because the currently widely used SHA1 is nearing the end of its useful life. The problem is that older clients might not support ECDSA keys and SHA2 signatures. One solution is to deploy with two sets of keys and certificates: RSA and SHA1 for older clients and ECDSA and SHA2 for newer clients.

Deploying a site with multiple keys is straightforward: simply specify multiple keys and certificates, one set after another. For example:

```
# RSA key.
SSLCertificateKeyFile conf/server-rsa.key
SSLCertificateFile conf/server-rsa.crt

# DSA key.
SSLCertificateKeyFile conf/server-dsa.key
SSLCertificateFile conf/server-dsa.crt

# ECDSA key.
```

```
SSLCertificateKeyFile conf/server-ecdsa.key
SSLCertificateFile conf/server-ecdsa.crt

# Intermediate certificates; must work
# with all three server certificates.
SSLCertificateChainFile conf/chain.pem
```

The only catch is that the `SSLCertificateChainFile` directive can be used only once per server, which means that the intermediate certificates must be identical for all three certificates. There are early indications that the CAs who are starting to offer ECDSA keys are set up this way.

It's possible to use different certificate hierarchies, but then you must avoid `SSLCertificateChainFile` altogether. Instead, concatenate all the necessary intermediate certificates (for all the keys) into a single file, and point to it using the `SSLCACertificateFile` directive. There might be a slight performance penalty with this approach because, on every new connection, OpenSSL now needs to examine the available CA certificates in order to construct the certificate chain.

> **Note**
>
> To ensure that all deployed keys are actually used, make sure you also enable the corresponding cipher suites in the configuration. ECDSA suites have the word "ECDSA" in the name; DSA suites have the word "DSS" in the name; all other authenticated suites are designed to work with RSA keys.

Wildcard and Multisite Certificates

If you have two or more sites that share a certificate, it is possible to deploy them on the same IP address, despite the fact that virtual secure hosting is not yet feasible for public web sites. No special configuration is required; simply associate all such sites with the same IP address and ensure that they are all using the same certificate.[1]

This works because TLS termination and HTTP host selection are two separate steps. When terminating TLS, in the absence of SNI information (see the next section for more information) Apache serves the certificate of the default site for that IP address, which is the site that appears first in the configuration. In the second step, Apache looks at the `Host` request header provided and serves the correct site at the HTTP level. If the requested hostname is not configured on the IP address, the default web site will be served.

With this type of deployment, you might get a warning similar to this one:

[1] Technically, the restrictions are per IP address and port combination (a TCP/IP endpoint). You could, for example, host one secure site on `192.168.0.1:443` and another on `192.168.0.1.:8443`. In practice, public sites can be hosted only on port 443, so the restrictions are effectively per IP address.